Konzepte intuitiv verständlicher Systembeschreibungen

1. Gegen eine Überbewertung von Formalismen

Wenn eine Sache überbewertet wird, dann geschieht dies zwangsläufig zu Lasten von etwas anderem. Mit den folgenden Überlegungen soll die Behauptung gerechtfertigt werden, daß in der Informatik die Überbewertung von Formalismen zu Lasten der Komplexitätsbeherrschung geht.

Die Systeme der Informatik sind heute schon sehr komplex und werden in Zukunft noch sehr viel komplexer sein. Diese Systeme entstehen durch Entwurfsentscheidungen von Menschen, und Komplexitätsbeherrschung liegt nur vor, wenn alle Entwurfsentscheidungen begründet und akzeptiert werden können. Dabei sind es wieder Menschen, denen gegenüber etwas begründet werden muß und die etwas akzeptieren können. Deshalb hängt Komplexitätsbeherrschung sehr stark von der Fähigkeit von Menschen ab, effizient miteinander zu kommunizieren.

Jedes Informatiksystem ist ein System zur Abwicklung von Formalismen, die einem bestimmten Zweck dienen. Dabei ist ein Formalismus eine Menge von Vorschriften zur Transformation von Formen. Aus einer vorgegebenen Form werden nach vorgegebenen Regeln neue Formen erzeugt, wobei die Anwendung der Regeln keine Zuordnung von Bedeutungen zu den Formen erfordert. Die Wahl der Formen und der Transformationsregeln trifft der Mensch im Hinblick auf den beabsichtigten Zweck.

Die Menschen, die sich beruflich mit Formalismen befassen, lassen sich grob in zwei Klassen einteilen je nachdem, ob ihr primäres Interesse dem Formalismus oder dem Zweck gilt. Man kann die beiden Klassen überspitzt wie folgt charakterisieren: In der einen Klasse befinden sich all diejenigen, die zuerst einen Formalismus erfinden und analysieren und danach erst nach einer nützlichen Anwendung dafür suchen; in der anderen Klasse findet man all diejenigen, die vor einem praktischen Problem stehen und dafür einen nützlichen Formalismus suchen. Bei den Mitgliedern der ersten Klasse dominiert das mathematische Interesse, wogegen bei den Mitgliedern der zweiten Klasse das Ingenieursinteresse dominiert.

Bei der Behandlung komplexer Informatiksysteme muß man selbstverständlich mit der Betrachtung des Zwecks beginnen und im Laufe der Betrachtung zu den Formalismen übergehen. Eine Überbewertung der Formalismen liegt vor, wenn bei der Betrachtung zu früh auf die Formalismen übergegangen wird, und entsprechend liegt eine Unterbewertung der Formalismen vor, wenn man bei der Betrachtung zu spät auf die Formalismen übergeht. Dabei bedeutet der Übergang zur Betrachtung von Formalismen, daß man die Formen und deren Transformationen in einer solchen Weise betrachtet, daß dabei mit den betrachteten Elementen und Strukturen kein intuitives Verständnis mehr verbunden sein muß.

Dies bedeutet selbstverständlich nicht, daß man mit der Verwendung formaler Begriffe erst beginnen sollte, wenn man in die formale Betrachtung eintritt. Auch in dem Teil einer Systembetrachtung, worin es um den Zweck und damit um ein intuiti-

ves Systemverständnis geht, dient es der sprachlichen Präzision und damit dem Kommunikationswirkungsgrad, wenn man Begriffe aus der formalen Welt mit anschaulicher Interpretation verwendet. Es ist hier insbesondere an die Begriffswelt der Mengenlehre und der formalen Logik gedacht.

Die Wahlmöglichkeit, sein primäres Interesse auf die Formalismen oder deren Zweck zu richten, findet ihre Entsprechung in zwei Fragen, die Dijkstra in [ ] zur Charakterisierung zweier streng auseinanderzuhaltender Problembereiche formuliert hat. Die eine Frage lautet: Ist das spezifizierte System das gewollte? Und die andere Frage lautet: Erfüllt das realisierte System die Spezifikation? Mit der ersten Frage wird das *Validierungsproblem* charakterisiert, mit der zweiten Frage das *Verifikationsproblem*. Das Verifikationsproblem läßt sich als mathematisches Problem behandeln, wenn die formale Spezifikation vollständig ist, d.h. wenn bei der Interpretation der Spezifikation ganz auf ein intuitives Verständnis verzichtet werden kann. Bisher wurde noch kein wirklich komplexes System in diesem Sinne vollständig formal spezifiziert, und es ist auch nicht abzusehen, ob dies jemals geschehen wird. Dies ist aber keine Frage der grundsätzlichen Machbarkeit, sondern eine Frage der Wirtschaftlichkeit.

Im Gegensatz zum Verifikationsproblem läßt sich das Validierungsproblem grundsätzlich nicht zu einem mathematischen Problem machen. Es geht ja hierbei um die Frage, ob das spezifizierte System das gewollte sei. Die Information über das Gewollte befindet sich aber eingeschlossen in den Köpfen von Menschen. Wenn sie dieses Gewollte in Schrift und Zeichnung formulieren, können sie schon nicht mehr sicher sein, ob das Geschriebene und Gezeichnete tatsächlich das ausdrückt, was sie wollten. Je anschaulicher ihre Formulierungsmittel sind, mit denen sie ihr Gewolltes ausdrücken, desto leichter wird es ihnen fallen, das Formulierte daraufhin zu überprüfen, ob es mit dem Gewollten übereinstimmt. Je unanschaulicher die Formulierungsmittel sind, um so eher kann es vorkommen, daß der Wollende etwas formuliert, von dem er glaubt, es beschreibe das Gewollte, während es tatsächlich vom Gewollten abweicht. Nicht jede Sprache eignet sich deshalb in gleicher Weise für die Formulierung des Gewollten.

Normalerweise wird eine Sprache, die zur Bewältigung des Validierungsproblems angemessen ist, nicht auch für die Bewältigung des Implementierungsproblems angemessen sein. Das *Implementierungsproblem* besteht darin, für das Gewollte eine Formulierung zu finden, die man dem Computer übergeben kann, damit dieser dann zum gewollten System wird. Dem Computer muß man selbstverständlich eine vollständig formale Formulierung übergeben, denn dieser kennt keine intuitive Semantik. Für den Computer ist es nur wichtig, daß die Formulierung des Gewollten formalsemantisch vollständig und widerspruchsfrei ist.

Aus diesen Überlegungen folgt, daß man mehrere Formulierungen benötigt, um von dem Willen, ein System mit bestimmten Eigenschaften zu realisieren, zu dem tatsächlich realisierten System zu gelangen. An einem sehr einfachen Beispiel soll die Notwendigkeit solcher unterschiedlicher Formulierungen veranschaulicht werden.

Bei diesem Beispiel geht es gar nicht um ein komplexes System, sondern um einen einfachen Stapel. Als Sprache, bei deren Verwendung man ziemlich sicher sein kann, daß das Formulierte mit dem tatsächlich Gewollten übereinstimmt, benutzt man im gegebenen Fall zweckmäßigerweise die Begriffswelt der Mengenlehre und verbindet diese mit der Anschauung.

Da das Wort "Stapel" in drei verschiedenen Bedeutungen benutzt wird, sollte man sorgfältig darauf achten, daß bei jeder Verwendung dieses Wortes eindeutig klar ist, in welcher Bedeutung es jeweils aktuell benutzt wird. Mit dem Wort "Stapel" kann eine Speichervariable gemeint sein oder ein aktueller Speicherinhalt oder ein System, welches aus einem Speicher und einem Akteur besteht, der von seiner Umgebung Anfragen oder Änderungswünsche bezüglich des Speicherinhalts entgegennimmt und diese beantwortet bzw. ausführt.

Wenn man von einem Stapel im Sinne eines Speicherinhalts spricht, meint man damit eine linear geordnete endliche Menge, und man verbindet damit die Anschauung einer Menge von flachen Gegenständen, die man auf einer horizontalen Fläche zu einem Stapel aufgetürmt hat. Deshalb bezeichnet man das in der Linearordnung am einen Ende liegende Element als das unterste und das am anderen Ende der Ordnung liegende Element als das oberste.

Wenn man von einem Stapel im Sinne eines Systems spricht, welches aus einem Speicher und einem darauf zugriffsberechtigten Akteur besteht, verbindet man mit dem Akteur ein ganz bestimmtes Repertoire von Anforderungen, die er von der Systemumgebung entgegennehmen kann. Es wird hier das folgende Repertoire aus fünf Anforderungen betrachtet:

CLEAR	Nach Ausführung dieser Anweisung soll der Speicher leer sein, d.h. der Stapel soll anschließend kein einziges Element mehr enthalten.
PUSH(Element)	Das als Parameter dieser Anforderung übergebene Element soll als neues oberstes auf den bisherigen Stapel gelegt werden.
POP	Das aktuell oberste Element soll entfernt werden. Diese Anweisung ist unzulässig, wenn der aktuelle Stapel leer ist.
HEIGHT	Diese Anfrage soll mit der Angabe beantwortet werden, wieviele Elemente aktuell im Speicher liegen.
TOP	Diese Anfrage soll mit der Angabe beantwortet werden, welches Element aktuell als oberstes im Speicher liegt. In gleicher Weise wie die Anweisung POP ist auch die Anweisung TOP unzulässig, wenn der aktuelle Stapel leer ist.

Man beachte, daß es sich bei der gegebenen Beschreibung des Anforderungsrepertoires für das Stapelsystem nicht um eine formale Beschreibung dieses Systems handelt, obwohl in dieser Beschreibung die präzise Begriffswelt der Mengenlehre verwendet

wird. Denn bei der Angabe der Bedeutung der verschiedenen Anforderungen mußte auf die Anschauung einer vertikal im Raum liegenden, geordneten endlichen Menge von Dingen Bezug genommen werden. Nun aber soll versucht werden, eine formale Spezifikation des Stapelsystems zu skizzieren.

Die formale Spezifikation wird hier nicht vollständig ausgeführt; es geht lediglich darum, dem Leser das grundsätzliche Prinzip der formalen Spezifikation zu vermitteln. Dabei ist die hier gewählte Form der formalen Spezifikation nicht die einzig mögliche; es gibt durchaus unterschiedliche Ansätze zur formalen Spezifikation von Systemen. Der hier gewählte Ansatz wurde gewählt, weil er sich für den verfolgten Zweck besonders gut eignet. Es soll nämlich gezeigt werden, daß ein Mensch, der noch keine intuitive Vorstellung von dem spezifizierten System hat, aus einer formalen Spezifikation, die sich auf keinerlei Intuition stützt, keine intuitive Vorstellung gewinnen kann.

An dieser Stelle muß noch etwas mehr über die Rolle der Intuition im Bereich der formalen Beschreibungen gesagt werden. Eine formale Beschreibung muß selbstverständlich immer Beschreibungselemente verwenden, deren Bedeutung dem Leser schon bekannt ist, bevor er mit dem Lesen der Beschreibung beginnt. Mit diesen Elementen muß der Leser also eine intuitive Semantik verbinden. Neben diesen Elementen enthält eine formale Beschreibung nur noch Bezeichner für Elemente, die der eigentliche Gegenstand der Beschreibung sind, d.h. deren Bedeutung erst durch die Beschreibung selbst festgelegt wird.

Diejenigen Elemente einer Beschreibung, mit denen man schon vor dem Lesen vertraut sein muß, weil ihre Bedeutung nicht aus der Beschreibung selbst hervorgeht, sind die Elemente der Beschreibungsmethodik; die anderen Elemente dagegen, deren Bedeutung erst durch die Beschreibung selbst vermittelt werden soll, sind die Elemente des Beschriebenen. Die Elemente der Beschreibungsmethodik bilden eine sog. *Metawelt*; die Elemente des Beschriebenen bilden die sog. *Gegenstandswelt.*

Weiter oben wurde gesagt, daß sich eine formale Beschreibung nicht auf intuitive Anschauung stützen darf. Diese Aussage muß nun dahingehend präzisiert werden, daß der Verzicht auf intuitive Anschauung nur für die Gegenstandswelt, nicht aber für die Metawelt gelten kann.

Durch die nun anschließende Betrachtung einer formalen Spezifikation des Stapelsystems werden die zuletzt gemachten Aussagen über die Notwendigkeit der Unterscheidung zwischen der Metawelt und der Gegenstandswelt für den Leser verständlicher werden. Im betrachteten Beispiel besteht die Gegenstandswelt aus dem Stapelsystem und insbesondere aus dem Repertoire der Anforderungen an dieses System. Die zentralen Begriffe der Metawelt, die im folgenden erklärt werden sollen, sind: die Anforderungsfolge, das Legalitätsprädikat, die Folgenäquivalenz und die Ausgabefunktion.

In der hier vorzustellenden Metawelt geht es ausschließlich um die Beschreibung von Gegenstandswelten, die jeweils aus einem sequentiell zu betreibenden System beste-

hen, welches nacheinander auf Anforderungen reagiert, die aus einem endlichen Repertoire stammen. Deshalb ist die Anforderungsfolge ein zentraler Begriff der Metawelt. Jede *Anforderungsfolge* hat eine endliche Länge und ist als eine mögliche Folge von Anforderungen zu verstehen, die das System seit seiner Inbetriebnahme bis zum aktuellen Zeitpunkt entgegengenommen hat. Da nicht unbedingt in jedem Zustand des Systems jede Anforderung aus dem Repertoire akzeptiert werden kann, läßt sich bezüglich der Anforderungsfolgen das *Legalitätsprädikat* definieren. Eine Anforderungsfolge ist legal, wenn sie von dem System vollständig akzeptiert werden kann.

Zwei Anforderungsfolgen sind einander *äquivalent*, wenn sie bezüglich der Festlegung des zukünftigen Systemverhaltens austauschbar sind. Die Austauschbarkeit bezüglich der Festlegung des zukünftigen Systemverhaltens liegt genau dann vor, wenn für jede beliebige Folgenverlängerung die Reaktionen des Systems unabhängig davon sind, an welche der beiden äquivalenten Folgen die Verlängerung angehängt wird.

Der Argumentbereich der *Ausgabefunktion* ist die Menge aller legalen Anforderungsfolgen, die nicht die Länge Null haben. Die Ausgabefunktion ordnet jeder solchen Anforderungsfolge dasjenige Ausgabeelement zu, das von dem System als Reaktion auf die letzte Anforderung in der Folge geliefert wird.

Im betrachteten Fall äußert sich die Metawelt durch die folgenden Symbole:

- λ ist das Legalitätsprädikat.
- a ist eine Variable, die jeweils mit einer Anforderungsfolge zu belegen ist.
- $\bullet$ ist der Operator zum Aneinanderhängen zweier Teilfolgen.
- $\equiv$ symbolisiert die Äquivalenz der links stehenden Folge mit der rechts stehenden Folge.
- ω ist die Ausgabefunktion.

Wenn man die Bedeutung der Symbole für die Elemente der Metawelt kennt, liest man die nachfolgenden formalen Ausdrücke, die einen Auszug aus der formalen Spezifikation des Stapelsystems darstellen, verhältnismäßig leicht. Dies liegt daran, daß man durch die Bezeichnungen für die Elemente aus der Gegenstandswelt darauf hingewiesen wird, daß es sich hier um eine formale Beschreibung eines Stapelsystems handelt, von dem man bereits eine intuitive Vorstellung hat. Diese intuitive Vorstellung entsteht nicht erst durch die formale Beschreibung, sondern entstand bereits anläßlich einer nichtformalen Beschreibung, wie sie weiter oben anläßlich der Einführung des Anforderungsrepertoires gegeben wurde.

$\lambda[\, a \,] \rightarrow \lambda[\, a \bullet PUSH(Element) \,]$

$\lambda[\, a \bullet POP \,] \equiv \lambda[\, a \bullet TOP \,]$

$a \bullet HEIGHT \equiv a$

$a \bullet PUSH(Element) \bullet POP \equiv a$

$\lambda[\, a \bullet TOP \,] \rightarrow (\, a \bullet TOP \equiv a \,)$

$\omega[\, a \bullet PUSH(Element) \bullet TOP \,] = Element$

$\omega[\, a \bullet PUSH(Element) \bullet HEIGHT \,] = 1 + \omega[\, a \bullet HEIGHT \,]$

$\omega[\, HEIGHT \,] = \omega[\, a \bullet CLEAR \bullet HEIGHT \,] = 0$

Daß eine formale Beschreibung tatsächlich nicht geeignet ist, ein intuitives Verständnis für die Gegenstandswelt zu vermitteln, konnte der Autor in vielen Experimenten empirisch nachweisen. Er hat dazu die obigen Ausdrücke der formalen Spezifikation derart umgeschrieben, daß er anstelle aller auf den Stapel hinweisenden Bezeichner andere Bezeichner eingesetzt hat, mit denen man a priori keinerlei Anschauung verbindet. So wurden anstelle der Bezeichner CLEAR, PUSH, POP, HEIGHT und TOP die Bezeichner RAL, LING, HUM, HOOM und SES eingesetzt. Die derart modifizierte formale Spezifikation blieb praktisch allen Versuchspersonen unverständlich, und der Überaschungseffekt war stets verhältnismäßig groß, wenn man ihnen die Rückübersetzung der Bezeichner verriet.

Formale Beschreibungen sind zwar das beste Mittel zur Lösung des Mißverständlichkeitsproblems; das betrachtete Beispiel zeigt aber, daß sie nicht geeignet sind, das Unverständlichkeitsproblem zu lösen. Das *Mißverständlichkeitsproblem* besteht in der Möglichkeit, daß jemand beim Lesen einer Beschreibung glaubt, alles verstanden zu haben, was der Schreiber sagen wollte, der Schreiber aber dennoch etwas anderes gemeint hat. Das *Unverständlichkeitsproblem* dagegen besteht darin, daß der Leser eine Beschreibung zwar als Form zur Kenntnis nehmen kann, ihr aber keine schlüssige Information entnehmen kann. Man benötigt deshalb immer zwei Arten von Beschreibungen. Zur Vermittlung eines intuitiven Systemverständnisses braucht man eine nichtformale Beschreibung, und zur Korrektur möglicher Mißverständnisse und Informationsdefizite, welche die nichtformale Beschreibung möglicherweise hinterläßt, braucht man anschließend eine formale Beschreibung, bei der man aber die Bezeichner so wählen muß, daß der Bezug zum intuitiven Verständnis erhalten bleibt.

Es dürfte leicht einzusehen sein, daß man sich unnötigerweise eine Menge schwieriger Probleme einhandelt, wenn man der Erstellung von Beschreibungen zur Vermittlung des intuitiven Systemverständnisses nicht die erforderliche Sorgfalt widmet. Selbstverständlich kann man trotz aller Sorgfalt keine guten Ergebnisse erzielen, wenn man nicht weiß, wie man methodisch an die Erstellung solcher Beschreibungen

herangehen soll. Deshalb ist es eine zentrale Aufgabe der Informatik als Ingenieursdisziplin, geeignete Regeln und Methoden zu suchen und zu vermitteln, deren Beachtung bzw. Anwendung sicher stellt, daß Beschreibungen angefertigt werden, die dem Bedürfnis nach einem hohen Kommunikationswirkungsgrad gerecht werden. Leider hat die Informatik als wissenschaftliche Disziplin diese Aufgabe bisher sträflich vernachlässigt. Fast jede beliebige Stichprobe, bei der man irgendwelche Lehrbücher, wissenschaftliche Aufsätze oder Software–Handbücher studiert, offenbart den betrüblichen Sachverhalt, daß das methodische Beschreiben von Systemen zur Vermittlung eines intuitiven Systemverständnisses nicht nur nicht beherrscht, sondern noch nicht einmal ernsthaft versucht wird.

In der Informatik herrscht eine derart übermäßige Begeisterung für Formalismen und damit für Automatismen vor, daß dadurch die Wahrnehmung wesentlicher Probleme verhindert wird, die gelöst werden müßten, damit die Informatik zu einer Ingenieurwissenschaft wird. Es wird hier ja keineswegs verlangt, daß die Informatik sich nicht mehr mit Formalismen befassen solle. Es wird lediglich behauptet, daß sich die Informatiker bei der Behandlung von Systemen voreilig in die Formalisierung stürzen und sich dadurch unnötige, aber schwerwiegende Probleme einhandeln. Es gibt sogar Fälle, wo Informatik–Autoren stolz über die Lösung von Problemen berichten, die sie gar nicht gehabt hätten, wenn sie nicht zu früh formalisiert hätten. In diesen Fällen hat die voreilige Formalisierung das Erkennen weitgehender Vereinfachungsmöglichkeiten verhindert. Ein typischer solcher Fall ist die Behandlung des Pseudoproblems riesiger Zustandszahlen, welches im Abschnitt 2 skizziert wird.

Die Schwerpunktsetzung auf die Formalismen und die Vernachlässigung der Erfordernisse der zwischenmenschlichen Kommunikation kommen auch in den Schriften zum Ausdruck, in denen graphische Darstellungsmittel wie Petrinetze oder Statecharts eingeführt werden. In diesen Einführungen werden Petrinetze oder Statecharts primär als Formalismen eingeführt und nicht als Darstellungsmittel zur Gestaltung von Beschreibungen mit hohem Kommunikationswirkungsgrad. Wenn ein Autor ein Petrinetz als Quintupel einführt, für das er anschließend eine graphische Darstellungsmöglichkeit zeigt, dann geht es ihm offensichtlich primär um den Formalismus und nicht um die Nutzung der Gestaltwahrnehmungsfähigkeiten des Menschen. Auch das Layout der Petrinetze oder Statecharts, die von den Autoren als Beispiele gezeigt werden, ist in den meisten Fällen derart unbefriedigend, daß man sicher sein kann, daß die Autoren der Optimierung des Layouts keinerlei Aufmersamkeit gewidmet haben. Das Interesse solcher Autoren geht in die gleiche Richtung wie bei Graphentheoretikern, die dicke Bücher über ihr Fach schreiben, worin keine einzige Graphik vorkommt.

Zum Abschluß dieses Abschnitts über die Überbewertung von Formalismen will der Autor eine Begebenheit berichten, die er auf einem Softwaretechnologiekongress erlebt hat und die das in den hier vorgestellten Überlegungen behandelte Problem kurz und prägnant vor Augen führt. In der Diskussion, die sich an die Präsentation des Beitrags [Zuck] auf diesem Kongress anschloß, sagte eine Teilnehmerin: "Aus Ihrer Darstellung von Methoden der Systembeschreibung habe ich nicht erkannt, wie man

diese Beschreibungen zur automatischen Generierung von Programmcode nutzen könnte. Wenn man daraus aber keinen Programmcode generieren kann, sind Ihre Beschreibungen doch nutzlos." Deutlicher hätte diese Dame ihre Einäugigkeit nicht offenbaren können.

2. Der Zustandsbegriff und sein Umfeld

In der Systemtheorie wurden der Begriff des Systemzustands und damit die Zustandsvariablen eingeführt, damit man das Systemverhalten in einer bestimmten einleuchtenden Form beschreiben konnte. In der Systemtheorie ging es primär um Kontinuumssysteme, also um Systeme, in denen an den verschiedenen Beobachtungsorten in jedem Zeitpunkt definierte Werte zu finden sind. Man erkannte, daß man die Variablen innerhalb eines Systems in zwei Klassen einteilen kann in Abhängigkeit von der Frage, ob die physikalischen Zwänge für die jeweilige Variable eine sprunghafte Wertänderung zulassen oder nicht. In denjenigen Variablen, bei denen sprunghafte Wertänderungen aus physikalischen Gründen unmöglich sind, äußern sich bestimmte Trägheitseffekte. Diese Trägheitseffekte beruhen darauf, daß sich Materie oder Energie nicht beliebig schnell von einem Ort zu einem anderen transportieren läßt. Deshalb muß beispielsweise der Füllstand eines Wasserbehälters eine Zustandsvariable sein ebenso wie die Spannung eines elektrischen Kondensators oder die Durchbiegung einer elastischen Blattfeder.

Durch Abstraktion gelangte man von den Trägheitseffekten zum Begriff des Gedächtnisses. Man konnte nun die aktuelle Belegung einer Zustandsvariablen als Information über die Vergangenheit deuten. Der jeweils aktuelle Zustand stellt dann dasjenige Wissen dar, welches das System benötigt, um sich jetzt und in Zukunft gemäß seiner Spezifikation verhalten zu können. Umgekehrt kann man also sagen, eine Spezifikation könne derart sein, daß sie das System nur erfüllen kann, wenn es in jedem Zeitpunkt noch über eine gewisse Information aus der Vergangenheit verfügt. Im Sinne der Systemtheorie ist also nicht jede innere Variable eines Systems, an deren Wert man interessiert ist, eine Zustandsvariable.

Da sich der Zustandsbegriff, wie ihn die Systemtheorie eingeführt hat, seit langem bewährt hat und man keinen zwingenden Grund angeben kann, weshalb man bei der Betrachtung von Systemen der Informatik einen anderen Zustandsbegriff bräuchte, wäre es eine völlig unnötige Kommunikationserschwernis, wenn man in Informatiktexten das Wort *Zustand* in einem anderen Sinne benutzen würde.

In Systemen der Informatik spielen die stetig veränderlichen Zustandsvariablen kaum eine Rolle; vielmehr geht es hier um Variable mit diskreten Wertebereichen. Die Diskretheit dieser Wertebereiche entsteht dabei immer durch Quantisierung eines physikalischen Kontinuums. So wird beispielsweise bei einem Zahnrad der kontinuierliche Umfang durch die Zähne quantisiert, oder es wird der kontinuierliche Kontaktweg eines Lichtschalters durch seine beiden Endstellungen quantisiert, wodurch auch das Kontinuum der Beleuchtungsintensität des Raumes in die beiden Ex-

tremwerte quantisiert wird. Wegen der durch die Quantisierung herbeigeführten Diskretheit der Wertebereiche kann man diese Zustandsvariablen nicht mehr durch die Frage finden, ob bei ihnen aus physikalischen Gründen sprunghafte Wertänderungen unmöglich seien; denn durch die Quantisierung hat man ja bereits die Ebene der physikalischen Betrachtung verlassen. Bei Variablen mit diskreten Wertebereichen ist jede Wertänderung ein Sprung. Innerhalb des Kontinuums, in dem die Quantisierung erfolgte, kann der Übergang von einem diskreten Wert zum anderen zwar nicht sprunghaft erfolgen, aber dieser kontinuierliche Übergang wird für die Betrachtung der Wertewechsel im Diskreten zwangsläufig irrelevant.

Bei den Kontinuumssystemen kann man auf die Vorstellung verzichten, daß die Zustandsvariablen das Gedächtnis des Systems darstellen, weil man hier die Zustandsvariablen über die Frage nach der Sprungfähigkeit findet. Im Gegensatz dazu benötigt man bei der Betrachtung diskreter Systeme unbedingt die Vorstellung vom Systemgedächtnis, denn hier ist die Frage nach der Sprungfähigkeit keine Hilfe bei der Suche nach den Zustandsvariablen. In diskreten Systemen ist eine Zustandsvariable also immer einem Speicher für diskrete Werte zugeordnet.

Nicht jede Variable in einem diskreten System ist zwangsläufig eine Zustandsvariable, denn nicht jeder Variablen, deren Belegung beobachtet werden kann, ist ein Speicher zugeordnet. Als Beispiel betrachte man das bereits erwähnte System aus einem Lichtschalter und einer Lampe: Der Lichtschalter, den man per Hand in die eine oder in die andere Stellung bringen kann, in der er verbleibt, nachdem man die Hand weggenommen hat, ist ein Speicher für einen Binärwert. Die Lampe dagegen, die man auch als binäre Variable im System erlebt, ist keinem Speicher zugeordnet, sondern ihr Wert ist bereits durch die Stellung des Lichtschalters vollständig festgelegt. Es wäre hier also falsch, die Intensitätsvariable des Lichtes als Zustandsvariable zu bezeichnen, falls man die Stellung des Lichtschalters als weitere Variable im System betrachtet. Wenn man jedoch auf die Betrachtung der Stellung des Lichtschalters völlig verzichtet und nur noch die Intensitätsvariable des Lichtes betrachtet, dann muß diese Intensitätsvariable als Zustandsvariable klassifiziert werden, denn sie ist in diesem Fall unmittelbar dem Binärspeicher zugeordnet.

Zustandswechsel, also Änderungen der Belegung von Zustandsspeichern, bedürfen jeweils eines Anstoßes. Ein Anstoß wird in der Systemtheorie idealisiert auf einen Zeitpunkt konzentriert, obwohl in der physikalischen Realität ein Anstoßvorgang eine endliche Dauer haben muß. Man denke an einen Startschuß, der eine endliche Dauer hat, aber in der Idealisierung auf einen Zeitpunkt gelegt werden kann. Ein System, welches in Ruhe ist, kann seinen Zustand nur ändern, wenn es von außen angestoßen wird. Ein externer Anstoß kann dann aber eine große Menge von internen Anstößen zur Folge haben. Eine Systembetrachtung mit gröbster Zeitauflösung bedeutet, daß man sich für die internen Anstöße nicht interessiert, sondern als Zustände des Systems nur die Ruhezustände akzeptiert. Bei dieser Betrachtung bleibt alles, was innerhalb des Systems nach einem externen Anstoß geschieht, bis das System wieder den nächsten Ruhezustand erreicht, im sog. Übergangsintervall verborgen.

Man kann nun selbstverständlich die zeitliche Auflösung der Betrachtung verfeinern und sich auch für Zustände interessieren, die innerhalb des Übergangsintervalls von einem Ruhezustand zum nächsten auftreten. Es hängt vom System ab, ob sich innerhalb des Übergangsintervalls eine kausalbedingte Sequenz von Zustandsübergängen abspielt oder ob auch sog. nebenläufige Zustandsübergänge vorkommen können. Im Falle von Nebenläufigkeit gibt es nicht eine einzige kausale Kette von Zustandsübergängen, sondern ein Geflecht von kausalen Ketten, die teilweise unabhängig voneinander ablaufen.

Wenn man Systemtheorie betreibt, muß man zwei Betrachtungsebenen streng auseinander halten, nämlich die Ebene der uninterpretierten Zustände und die Ebene der interpretierten Zustände. Die Betrachtung uninterpretierter Zustände dient dazu, bestimmte Begriffe einzuführen, die man anschließend bei der Betrachtung interpretierter Zustände als bekannt voraussetzt. So wird also nun im folgenden angenommen, daß die Bedeutung der Wörter Zustand, Zustandsübergang, Nebenläufigkeit oder Anstoß eindeutig festgelegt sei.

Die Zustandsvariablen eines Systems ergeben sich aus der Aufgabenstellung, die mit diesem System erfüllt werden soll. Jede Zustandsvariable dient ja dazu, eine Information zu speichern, die später noch gebraucht wird. Aus der Anzahl der verschiedenen Zustandsvariablen und der Mächtigkeit ihrer Wertebereiche ergibt sich für komplexe Systeme meistens eine riesige Zahl potentieller Zustände. Obwohl diese Zustandsmenge endlich ist, kommt eine explizite Aufzählung praktisch nicht in Frage. Manche Autoren haben die Beherrschung derart riesiger Zustandsmengen als formales Problem aufgefaßt, welches sie versucht haben zu lösen. Sie sind aber nur dadurch überhaupt zu ihrem Problem gekommen, daß sie die Interpretation der Zustandsvariablen außer Acht gelassen haben und deshalb alle Zustandsvariablen formal gleich behandeln mußten. Diese Vorgehensweise ist der Sache jedoch keineswegs angemessen. Wenn man nämlich die Interpretation der Zustandsvariablen mit in Betracht zieht, ergibt sich eine gewaltige Vereinfachung des Problems.

Es gibt grundsätzlich zwei unterschiedliche Typen von Zustandsvariablen, die man in den Beschreibungs– und Konstruktionsverfahren völlig unterschiedlich behandeln muß. Es handelt sich um die grundsätzliche Unterscheidung zwischen *Operationsvariablen* und *Steuervariablen*. Daß diese Unterscheidung eine gewaltige Vereinfachung des Problems großer Zustandsmengen mit sich bringt, wurde im Bereich der Automaten– und Schaltwerkstheorie bereits in der zweiten Hälfte der 60–iger Jahre von verschiedenen Autoren festgestellt und beschrieben [ ], [ ].

Ob eine Zustandsvariable als Operationsvariable oder als Steuervariable zu klassifizieren ist, entscheidet sich an der Frage, woher man den Wertebereich dieser Variablen kennt. Es werden nun Beispiele betrachtet, die diesen Unterschied veranschaulichen und die es leicht machen, den Unterschied zwischen Operationsvariablen und Steuervariablen in einer strengen allgemeinen Definition zu fassen. In einem ersten Beispiel werden die Ruhezustände eines stark vereinfachten fahrkartenverkaufenden Automaten betrachtet. Die starke Vereinfachung besteht zum einen darin, daß der

Automat nur eine einzige Art von Fahrkarten verkaufen kann, so daß der Käufer nicht angeben muß, welche Art Fahrkarte er haben will. Die Anstöße, die der Käufer gibt, bestehen lediglich in dem Einwurf von Münzen.

Eine weitere Vereinfachung der Betrachtung besteht darin, daß angenommen wird, der Automat besäße einen unerschöpflichen Vorrat an Fahrkarten und Wechselgeldmünzen, so daß die Füllung des Fahrkartenbehälters und die Füllung des Wechselgeldbehälters nicht in die Zustandsbetrachtung einzugehen brauchen. Der Automat befindet sich jeweils in einem Ruhezustand, wenn er auf den Einwurf einer Münze wartet. Da es unterschiedliche Münzen gibt und der Käufer frei entscheiden kann, welche Münzen er nacheinander einwirft, muß sich der Automat merken können, zu welchem Geldbetrag sich die in einem Kaufvorgang bisher eingeworfenen Münzen aufsummiert haben. Der Automat wird solange als Reaktion auf einen Münzeinwurf nichts ausgeben, bis die Summe der bisher eingeworfenen Münzen den Kaufpreis erreicht oder überschreitet. Erst dann wird er eine Fahrkarte und das sich gegebenenfalls aus einer Überzahlung ergebende Restgeld ausgeben. Der Automat benötigt also nur eine einzige Zustandsvariable, in der der jeweils erreichte Anzahlungsstand gespeichert werden kann. Bei dieser Variablen besteht eine enge semantische Bindung zwischen einer angemessenen Variablenbenennung und ihrem Wertebereich. Man wird die Variable beispielsweise "Anzahlungszustand" nennen, und daraus kann man bei Kenntnis des Münzrepertoires und des Fahrkartenpreises eindeutig auf den Wertebereich dieser Variablen schließen. Man braucht ja nur zu überlegen, welche unterschiedlichen Summen man mit dem gegebenen Münzrepertoire realisieren kann, die kleiner sind als der Fahrkartenpreis.

Die Variable "Anzahlungszustand" hat die beiden typischen Merkmale einer Operationsvariablen: Man findet ihren Wertebereich, ohne über eine Ordnung von Schritten in einem Algorithmus nachdenken zu müssen, und man kann die Interpretation der Elemente des Wertebereichs vollständig festlegen, ohne diese Elemente explizit aufzählen zu müssen. Der Sachverhalt, daß hier keinerlei Bedarf an einer expliziten Aufzählung der Elemente des Wertebereichs besteht, hat zur Konsequenz, daß auch eine extreme Erweiterung des Wertebereichs keinerlei Probleme mit sich brächte. Man stelle sich zum gegebenen Beispiel vor, der Fahrkartenpreis betrage eine Million DM und die kleinste Münze im Repertoire sei die 10–Pfennigmünze. In diesem Falle hätte der Wertebereich der Anzahlungsvariablen die Mächtigkeit 10^7, aber der Automat wäre dennoch nicht schwieriger zu verstehen, als wenn der Fahrkartenpreis nur 1,50 DM beträgt und somit die Mächtigkeit des Anzahlungswertebereichs nur 15 ist.

Als zweites Beispiel wird eine Digitaluhr betrachtet, bei der aber zur Vereinfachung auf manche Funktionen verzichtet wird, die zu einer Uhr normalerweise unbedingt dazu gehören. Es wird angenommen, diese Uhr enthalte jeweils einen Speicher für die aktuelle Uhrzeit, das Datum und die gewünschte Weckzeit. Es wird ferner angenommen, daß es keinerlei interne Anstöße zur Zustandsveränderung gebe; dies bedeutet, daß der Taktgeber, welcher normalerweise die aktuelle Uhrzeit fortschaltet,

nicht laufen soll und daß ferner die Übereinstimmung der aktuellen Uhrzeit mit der gewünschten Weckzeit keinerlei Konsequenzen haben soll.

Zur Beeinflussung der Speicherinhalte soll es zwei Druckknöpfe A und B geben, wobei man durch das Drücken von A darauf Einfluß nehmen kann, welche Wirkung das Drücken von B hat. Es soll vier unterschiedliche Wirkungsmöglichkeiten für den Druckknopf B geben: (1) Das Drücken von B bewirkt überhaupt nichts; (2) das Drükken von B erhöht die gespeicherte aktuelle Uhrzeit um eine Zeiteinheit; (3) das Drükken von B erhöht das gespeicherte aktuelle Datum um eine Datumseinheit; (4) das Drücken von B erhöht die gespeicherte Weckzeit um eine Zeiteinheit. Die hier mit (1) bis (4) durchnumerierten Fälle können jeweils durch das Drücken von A zyklisch durchlaufen werden.

Es ist nun ganz offensichtlich, daß es in dieser Uhr neben den drei Speichern für die aktuelle Uhrzeit, das Datum und die gewünschte Weckzeit noch einen vierten Zustandsspeicher geben muß, dessen Belegung darüber entscheidet, welche der vier möglichen Wirkungen bei einem Drücken von B aktuell eintritt. Bei dieser Zustandsvariablen handelt es sich eindeutig um eine Steuervariable. Sie hat nämlich zweifellos die beiden Merkmale, die für Steuervariable kennzeichnend sind: Man findet ihren Wertebereich nur durch explizite Aufzählung seiner Elemente, und man muß über eine Ordnung von Schritten in einem Algorithmus nachdenken, wenn man die Interpretation der Elemente dieses Wertebereichs vollständig erfassen will. Daß man im gegebenen Fall über eine solche Ordnung von Schritten nachdenken muß, erkennt man leicht daran, daß es ja nicht selbstverständlich ist, daß man durch Drücken der Taste A aus dem Fall (1) zum Fall (2) kommt; der Konstrukteur hätte ja auch entscheiden können, daß auf den Fall (1) zunächst einmal der Fall (3) folgt.

Da die einzelnen Elemente des Wertebereichs einer Steuervariablen explizit aufgezählt werden müssen und da außerdem explizit die möglichen Wertübergänge festgelegt werden müssen, ist der Zustandsgraph das geeignete Mittel zur Darstellung der Sachverhalte, die eine Steuervariable betreffen. Demgegenüber brächte es keinerlei Vorteile, den Wertebereich und die Wertübergänge einer Operationsvariablen in einem Zustandsgraphen zu erfassen, selbst wenn die Mächtigkeit des Wertebereiches dieses zuließe. Damit entpuppt sich das Problem der riesigen Zustandszahlen als ein Scheinproblem, denn die eigentliche Ursache für die großen Zustandszahlen liegt in den Mächtigkeiten der Wertebereiche der Operationsvariablen, die man aber gar nicht elementweise zu betrachten braucht. Der Sachverhalt, daß man den Wertebereich einer Steuervariablen gar nicht definieren kann, ohne seine Elemente explizit aufzuzählen, sorgt zwangsläufig dafür, daß die Mächtigkeit solcher Wertebereiche innerhalb beherrschbarer Grenzen bleibt.

In der Automaten– und Schaltwerkstheorie spielen Verfahren zur Reduktion von Zustandsmengen eine große Rolle. Die Frage nach einer möglichen Zustandsreduktion ist aber nur bezüglich der Steuervariablen sinnvoll; eine Operationsvariable kann von vorneherein aus semantischen Gründen nie Gegenstand einer Reduktionsüberlegung sein.

Während die Anzahl unterschiedlicher Operationsvariablen in einem System verhältnismäßig hoch sein kann, gibt es immer nur wenige unterschiedliche Steuervariable in einem System, häufig sogar nur eine einzige. Die Operationsvariablen entsprechen den frei deklarierten Variablen eines Programms, während die Steuervariable im Programm dem Befehlszähler entspricht, neben dem es normalerweise nicht noch weitere Befehlszähler gibt. Wenn in einem System mehrere Steuervariable vorkommen, dann ist es immer sinnvoll, dieses System in kommunizierende Teilsysteme zu zerlegen, von denen jedes jeweils nur eine Steuervariable enthält. Daß man auch sehr komplexe Systeme für das menschliche Verständnis gut faßbar darstellen kann, indem man das System als Netz aus kommunizierenden Teilsystemen darstellt, ist inzwischen in sehr vielen Fällen der Praxis demonstriert worden.

Alle hier gemachten Aussagen zur Klassifikation von Zustandsvariablen in Operations– und Steuervariable beziehen sich nur auf die sogenannten "primären Zustandsvariablen". Eine Zustandsvariable ist primär, wenn sie nicht erst durch eine Codierung entstanden ist; sonst nennen wir sie "sekundär". Die Elemente des Wertebereichs einer Zustandsvariablen müssen codiert werden, damit sie in technischen Systemen gespeichert und verknüpft werden können. Da aber der Wertebereich einer Zustandsvariablen schon bekannt sein muß, bevor man eine Codierung seiner Elemente festlegen kann, wird die Zustandsvariable vor ihrer Codierung als primär bezeichnet.

Normalerweise geschieht eine Codierung derart, daß den Elementen des primären Wertebereichs Codewörter zugeordnet werden, deren Stellenbelegungen aus einem sekundären Wertebereich stammen, dessen Mächtigkeit sehr viel kleiner ist als die des primären Wertebereichs. Meistens handelt es sich bei dem sekundären Wertebereich um die binäre Menge {0, 1}. Jede Variable für eine Stelle in einem Zustandscodewort ist eine sekundäre Zustandsvariable, und ihre Klassenzugehörigkeit – Operations– oder Steuervariable – richtet sich nach der Klassenzugehörigkeit der primären Zustandsvariable, die codiert wurde.

3. Erscheinungen auf Kanälen

Jeder Kanal dient einem Transport von einer Quelle zu einer Senke. Ein Kanal zwischen einer Quelle und einer Senke existiert genau dann, wenn das von der Quelle Eingespeiste zur Senke fließt, ohne daß es von der Quelle mit einer Adresse versehen wurde, und wenn es keine Überholvorgänge gibt, die bewirken würden, daß die Reihenfolge, in der die Senke das Eingespeiste empfängt, eine andere ist als die Reihenfolge, in der quellenseitig eingespeist wurde.

Bei der Betrachtung der Vorgänge, die sich in einem dynamischen System abspielen, interessiert man sich nicht nur für die Belegung der Zustandsspeicher und deren Änderungen, sondern auch für die Erscheinungen auf den Kanälen, welche die Verbindung zwischen dem System und seiner Umgebung sowie zwischen den Komponen-

ten innerhalb des Systems herstellen. Dabei gibt es sowohl für die Quelle als auch für die Senke jeweils zwei mögliche Sichten bezüglich ihrer Verbindung mit dem Kanal.

Für die Quelle ist der Kanal entweder ein *Anzeigekanal* oder ein *Paketkanal*. Damit ein Kanal Anzeigekanal sein kann, muß er bestimmte physikalische Eigenschaften haben. Eine Quelle kann einen Kanal nur dann als Anzeigekanal nutzen, falls die Senke durch den Kanal hindurch zur Quelle schauen kann, um zu sehen, welche Anzeigeinformation die Quelle am Kanaleingang bereithält.

Ein Kanal wird von einer Quelle als Paketkanal benutzt, wenn die Quelle dem Kanal ein räumlich und zeitlich begrenztes Materie– oder Energiepaket übergeben kann, welches dann über den Kanal von der Quelle wegfließt. Während ein Anzeigekanal nur der Informationsübertragung dienen kann, ist über einen Paketkanal nicht nur der Transport von Information, sondern auch der Transport von Materie möglich.

Die Frage, ob ein realer Kanal für eine Quelle ein Anzeigekanal oder ein Paketkanal sei, ist in vielen Fällen eindeutig entscheidbar; es gibt jedoch auch Fälle, wo diese Frage nicht allein auf Grund der beobachtbaren Vorgänge am Kanaleingang entschieden werden kann. In diesen zuletzt genannten Fällen wird die Kanalart lediglich durch den Zweck bestimmt, den die Quelle mit ihrem Einspeisen in den Kanal verbindet. Ein Anzeigekanal kann nur vorliegen, wenn es der Quelle grundsätzlich möglich ist, eine Anzeige am Kanaleingang sehr lange anzubieten. Wollte man beispielsweise einen akustischen Kanal als Anzeigekanal benutzen, dann würde man am Kanaleingang eine Anzeige jeweils in Form eines Dauertons bereitstellen, den man beliebig lange aufrechterhalten kann. Der gleiche akustische Kanal wird dagegen zu einem Paketkanal, wenn man ihn zur Übertragung eines Startschusses benutzt, der ja grundsätzlich eine sehr begrenzte Zeitdauer hat und nicht beliebig lange angeboten werden kann

Während es – wie gezeigt – möglich ist, ein und denselben akustischen Kanal einmal als Anzeigekanal und ein andermal als Paketkanal zu benutzen, gibt es diese Alternative bei Materialtransportkanälen nicht. Das anschaulichste Beispiel eines Materialtransportkanals, der zur Informationsübertragung benutzt wird, ist der Rohrpostkanal. Ein solcher Kanal besteht aus einem Rohr, durch welches mittels Druckluft eine Kapsel hindurchgedrückt wird, in die von der Quelle eine auf Papier geschriebene Botschaft gesteckt wurde. Ein solcher Rohrpostkanal kann nicht als Anzeigekanal, sondern nur als Paketkanal benutzt werden.

Als letztes Beispiel zur Unterscheidung zwischen Anzeige– und Paketkanälen wird nun ein Fall betrachtet, bei dem nach dem Zweck gefragt werden muß, den die Quelle mit der Kanalnutzung verbindet. Es wird der Kanal betrachtet, der einen potentiellen Fahrstuhlbenutzer mit dem technischen Fahrstuhlsystem verbindet. Dieser Kanal besteht aus einem Druckknopf und der Leitung, welche den Druckknopf mit der zentralen Steuerung verbindet. Grundsätzlich ist es möglich, daß der Benutzer sehr lange auf den Druckknopf drückt; dies bedeutet, daß der Kanal als binärer Anzeigekanal genutzt werden könnte. Andererseits wird der Benutzer normalerweise nur sehr kurz

der Konstruktion der Senke ab, ob der Kanal sendeseitig ein Anstoßkanal oder ein Angebotskanal ist. Die Fahrstuhlsteuerung könnte als getaktetes Schaltwerk realisiert sein, welches mit einer Frequenz von 100 kHz getaktet wird. In diesem Fall werden sämtliche Reaktionen der Senke durch den innerhalb der Senke liegenden Taktgenerator angestoßen, d.h. in diesem Fall gibt es keine Anstoßkanäle zwischen der Fahrstuhlsteuerung und ihrer Umgebung. Eine solche Fahrstuhlsteuerung schaut alle 10 Mikrosekunden auf das Ende des Angebotskanals, und in Abhängigkeit davon, was sie dort sieht, ist die durch den Takt angestoßene Reaktion von der einen oder der anderen Art. Die Vorderflanke des Tastendruckimpulses hat in diesem Fall keinerlei physikalische Reaktion in der Steuerung zur Folge. Anders liegt dagegen der Fall, wenn die Steuerung als ungetaktetes Schaltwerk realisiert wird. Dann nämlich ist es tatsächlich die Vorderflanke des Tastendruckimpulses, die über eine physikalische Kausalkette die Senkenreaktion auslöst.

4. Kausalordnung von Ereignissen

Nachdem nun ausführlich die Erscheinungen auf Kanälen betrachtet wurden, kann der Begriff der *Nebenläufigkeit* behandelt werden, der für die Betrachtung dynamischer Systeme unbedingt gebraucht wird. Dieser Begriff kann nur unter Bezug auf die Erscheinungen auf den Kanälen definiert werden. Komplementär zum Begriff der Nebenläufigkeit ist der Begriff der *Sequentialität*. Bei beiden Begriffen geht es um die Frage nach der kausalen Ordnung von Ereignissen. Die Ereignisse werden an den Enden der Kanäle beobachtet. Es gehört zum Wesen eines Kanals, daß er die an seinem Eingang auftretenden Ereignisse ordnungserhaltend zum Ausgang transportiert. Entsprechend der beiden alternativen Sichten auf der Quellenseite findet man dort als Ereignisse die Änderung einer Anzeige bzw. das Einspeisen einer Sendung in einen Paketkanal. Auf der Senkenseite findet man als Ereignisse die Änderungen des Angebots bzw. die Anstöße.

Da auf einem Kanal keine Ereignisse entstehen können und die Ordnung von Ereignissen nicht verändert werden kann, kann man sich bei der Betrachtung der Ordnung von Ereignissen in einem System pro Kanal jeweils auf eines seiner beiden Enden beschränken. Bezüglich der Kanäle, über die das System mit seiner Umgebung verbunden ist, ist es jeweils vorbestimmt, welche Kanalenden man betrachten muß: Bei den Eingangskanälen des Systems muß man das senkenseitige Ende und bei den Ausgangskanälen das quellenseitige Ende betrachten.

Bei den innerhalb des Systems liegenden Kanälen steht man vor der Entscheidung, welches Kanalende man jeweils betrachten soll. Jeder innerhalb des Systems vorkommende Kanal verbindet eine Systemkomponente mit einer anderen. Weil es bei der Suche nach Kausalketten wichtig ist zu wissen, wie die Systemkomponenten auf Ereignisse reagieren, ist es zweckmäßig, bei den systeminternen Kanälen jeweils das senkenseitige Kanalende zu betrachten. Denn das Verhalten einer Komponente kann man nicht beschreiben, wenn man nicht weiß, ob der betrachtete Kanal für diese

Komponente ein Angebotskanal oder ein Anstoßkanal ist. Somit werden also sowohl auf der Eingangsseite als auch innerhalb des Systems als Ereignisse nur die Angebotsänderungen und die Anstöße betrachtet; nur auf der Ausgangsseite des Systems muß man Anzeigeänderungen und Sendeereignisse betrachten.

Wenn man die Ereignisordnung analysieren will, muß man sich selbstverständlich zuerst einmal festlegen, ob man Ereignisse innerhalb des Systems überhaupt in die Betrachtung einbeziehen will. Man kann das System ja auch als Blackbox betrachten, bei der man nur die Ereignisse auf der Eingangsseite und der Ausgangsseite beobachten kann. Wenn ein System in Ruhe ist, kann kein ausgangsseitiges Ereignis mehr auftreten, ohne daß zuvor eingangseitig ein Anstoß erfolgt.

Ein sequentieller Systembetrieb liegt vor, wenn die Systemumgebung derart festgelegt ist, daß sie nie zwei Anstöße gleichzeitig liefert und nie einen Anstoß liefert, wenn das System nicht in Ruhe ist. Der sequentielle Betrieb eines Systems schließt die Nebenläufigkeit von Ereignissen auf der Systemausgangsseite nicht aus. Wenn auf der Eingangsseite keine nebenläufigen Ereignisse auftreten, kann die Nebenläufigkeit von Ereignissen auf der Ausgangsseite nur die Konsequenz einer innerhalb des Systems entstandenen Nebenläufigkeit sein.

Die Nebenläufigkeit, die innerhalb eines Systems entsteht, muß streng unterschieden werden von der Nebenläufigkeit, die von der Systemumgebung erzeugt wird. Nebenläufigkeit innerhalb des Systems kann nur dann entstehen, wenn es im System eine Kausalkettenaufspaltung gibt. Der typische Fall einer Kausalkettenaufspaltung liegt vor, wenn von einem auf einem Kanal angelieferten Paket mehrere Kopien hergestellt werden, die in unterschiedliche Kanäle eingespeist werden.

Die Systemumgebung hat zwei unterschiedliche Möglichkeiten, Nebenläufigkeit zu verursachen. Die eine Möglichkeit besteht darin, daß sie über unterschiedliche Kanäle nebenläufige Anstöße liefert. Die andere Möglichkeit besteht darin, daß sie den zeitlichen Abstand zwischen aufeinanderfolgenden Anstößen, die sie über einen Kanal liefert, so kurz wählt, daß ein neuer Anstoß bereits kommt, bevor die Reaktion des Systems auf den vorangegangenen Anstoß abgeschlossen ist.

Ein System kann dann und nur dann als Automat modelliert werden, wenn es sequentiell betrieben wird. Dabei ist es für die Automatenmodellierung unerheblich, ob bei der Reaktion auf einen Anstoß Nebenläufigkeit im System entsteht. Die möglichen Ruhezustände des Systems bilden das Zustandsrepertoire des Automaten, und die möglichen unterschiedlichen Anstöße bilden das Eingaberepertoire. Die Elemente des Ausgaberepertoires des Automaten findet man, indem man fragt, welche partiell geordneten ausgangsseitigen Ereignismengen als Reaktion auf einen Anstoß vorkommen können. Ein Element des Ausgaberepertoires ist also im allgemeinen Fall eine Partialordnungsstruktur aus Ereignissen, die im Sonderfall zur leeren Menge degeneriert sein kann.

Falls man bei getakteten Systemen den Taktgenerator zur Systemumgebung rechnet, liegt ein sequentiell betriebenes System vor, welches jeweils durch die Taktereig-

nisse angestoßen wird und bei dem die Anstöße zeitlich so weit auseinanderliegen, daß der nächste Anstoß erst kommt, wenn die Reaktion des Systems auf den vorangegangenen Anstoß abgeschlossen ist. Wenn man dagegen den Taktgenerator als Komponente des Systems betrachtet, muß man das Blackbox–Verhalten des Systems modellieren, ohne auf die Taktung Bezug zu nehmen. In diesem Fall hängt die Modellierung ausschließlich davon ab, in welcher Sicht die Umgebung das System betreibt. Bei dieser Betrachtung ist die Taktung erst durch eine Implementierungsentscheidung in das System hineingekommen und hat mit der Systemfunktion, die in der Aufgabenstellung spezifiziert wurde, nichts zu tun. In der Sicht der Aufgabenstellung könnte also das System ein anstoßfreier oder ein anstoßbedürftiger Zuordner oder ein Automat oder ein nebenläufig betriebenes System sein.

Nebenläufig betriebene Systeme können letztlich immer nur aus sequentiell betriebenen Komponenten aufgebaut werden, die über Kanäle in geeigneter Weise miteinander kommunizieren.

Unabhängig davon, ob man eine Blackbox–Betrachtung anstellt und sich nur für die Ereignisse am Eingang und am Ausgang des Systems interessiert, oder ob man auch Ereignisse in die Betrachtung einbezieht, die im Innern des Systems zu beobachten sind, muß man zu Beginn einer Betrachtung kausaler Abhängigkeiten zwischen Ereignissen jeweils die Orte festlegen, an denen man die Ereignisse beobachten will, und für jeden Beobachtungsort muß man das Repertoire der Ereignistypen festlegen, die man in die Betrachtung einbeziehen will.

Bei der Darstellung der kausalen Abhängigkeiten zwischen Ereignissen geht es darum, die Menge aller möglichen Kausalfolgengeflechte, die sich beim Systembetrieb ergeben können, mit endlichem Beschreibungsaufwand zu erfassen. Wenn die Kausalität als Ordnungskriterium für eine Ereignismenge gewählt wird, dann ergibt sich jedes Element einer Kausalfolge, welches nicht am Anfang der Folge steht, als Wirkung des unmittelbar voranstehenden Elements. Im Falle von Nebenläufigkeit stehen die Ereignisse nicht in einer Vollordnung, sondern in einer Partialordnung. Jede Partialordnung ist ein Folgengeflecht, d.h. durch die Partialordnung wird eine Menge von Folgen definiert, die gemeinsame Elemente haben können. Die auf diese Weise "verflochtenen" Folgen bilden einen zyklenfreien gerichteten Graphen.

Die Elemente des Folgengeflechts sind die Ereignisse, die bei einem aktuellen Systembetrieb tatsächlich vorkommen und die man nicht per Festlegung von der Betrachtung ausgeschlossen hat. Das Folgengeflecht kann keine Elemente enthalten, die nicht irgendwelchen Ereignissen entsprechen, die beim aktuellen Systembetrieb vorgekommen sind. Bei den meisten Systemen der Praxis gibt es Alternativen für den Systembetrieb, und zwar Alternativen, die durch die Systemumgebung ausgewählt werden, und Alternativen, die innerhalb des Systems entschieden werden. Die Auswahl einer Alternative durch die Systemumgebung erfolgt durch die Wahl der Systembeeinflussung, also durch die Art und die Reihenfolge der Elemente, die von der Umgebung in die Eingabekanäle des Systems eingespeist werden. Daß es auch Alternativen des Systembetriebs gibt, die im Innern des Systems ausgewählt werden, liegt

zum einen daran, daß es alternative Anfangszustände des Systems gibt, und zum anderen daran, daß es innerhalb des Systems zu Wettläufen von Ereignissen kommen kann, die indeterministisch entschieden werden. Wenn ein Ereignis das Einspeisen zweier Ereignisse in getrennte Kanäle bewirkt und das zukünftige Systemverhalten davon abhängt, welches der beiden Ereignisse als erstes an seinem Kanalende ankommt, dann muß man Verhaltensalternativen betrachten, falls nicht ein zeitunabhängiges Laufzeitverhältnis der beiden Kanäle konstruktionsbedingt garantiert ist.

Die Systeme, die hier betrachtet werden, haben im allgemeinen kein durch die Aufgabenstellung festgelegtes Betriebsende. Man denke an eine Armbanduhr oder eine Fahrstuhlsteuerung. Diese Systeme haben zwar eine begrenzte Lebensdauer, aber das Erreichen des Zeitpunkts ihrer Außerbetriebnahme ist kein mit dem regulären Systembetrieb kausal verbundenes Ereignis. Deshalb sind bei solchen Systemen die Folgengeflechte, die zum regulären Systembetrieb gehören, potentiell unendlich. Diese potentielle Unendlichkeit des Systembetriebs bringt es mit sich, daß auch die Menge der Alternativen für den Systembetrieb im allgemeinen eine unendliche Menge ist, denn wenn das System potentiell unendlich lang betrieben werden kann und während dieses Betriebs immer wieder Entscheidungen bezüglich der Zukunft des Betriebs gefällt werden müssen, fallen potentionell unendlich viele Entscheidungen.

Es wurde weiter oben gesagt, daß die kausalen Abhängigkeiten zwischen Ereignissen, die beim Systembetrieb auftreten können, mit endlichem Beschreibungsaufwand erfaßt werden sollen. Nun kann diese Aufgabe dahingehend präzisiert werden, daß es darum geht, eine im allgemeinen unendliche Menge von im allgemeinen unendlichen Folgengeflechten mit endlichem Beschreibungsaufwand zu erfassen. Die Beschreibung der kausalen Abhängigkeiten zwischen Ereignissen muß also ein generatives Schema sein, welches es gestattet, durch Anwendung einfacher Abwicklungsregeln jedes mögliche Anfangsstück eines jeden Folgengeflechts, welches im Systembetrieb auftreten kann, zu generieren.

Es gibt eine Fülle sehr unterschiedlicher Arten generativer Schemata, die das Geforderte leisten. Für den Vergleich und die Bewertung unterschiedlicher Arten generativer Schemata gibt es selbstverständlich etliche unterschiedliche Kriterien. In der vorliegenden Betrachtung interessiert nur ein einziges dieser Kriterien, nämlich die Eignung für den menschlichen Betrachter, d.h. die Eignung zur raschen Vermittlung eines intuitiven Verständnisses der Zusammenhänge. Es muß also nach der Nutzung der Gestaltwahrnehmungsfähigkeit des Menschen gefragt werden, denn es geht um die Sichtbarmachung der Entstehung von Nebenläufigkeit, um die Sichtbarmachung der Synchronisationsvorgänge, d.h. des Aufeinander Wartens, und um die Sichtbarmachung der Entscheidungen, mit denen Alternativen des Systembetriebs ausgewählt werden.

Die von uns über viele Jahre angestellten empirischen Untersuchungen lassen die Behauptung zu, daß eine bestimmte Form von Petrinetzen, nämlich die sicher markierten Stellen–Transitionsnetze, für den beschriebenen Zweck optimal geeignet sind.

Die Praxis hat gezeigt, daß es zweckmäßig ist, in diesen Petrinetzen sowohl bezeichnete als auch unbezeichnete Transitionen zu verwenden. Die Bezeichnung ordnet einer Transition einen Beobachtungsort im System und einen Ereignistyp zu, der an diesem Ort möglich ist. Jedes Schalten einer bezeichneten Transition im Laufe der Netzabwicklung entspricht dem Auftreten eines Ereignisses des genannten Typs an dem genannten Ort. Die unbenannten Transitionen werden lediglich aus syntaktischen Gründen eingesetzt; ihrem Schalten entspricht kein Ereignis im System.

Stellen müssen beschriftet werden, um ihre Markenkapazität anzugeben, falls diese größer als eins ist. Stellen mit der Markenkapazität eins können mit Werten von Anzeigevariablen beschriftet werden, falls diese Wertbelegung eindeutig mit der Stellenmarkierung verbunden ist. Pfeile, die von einer Transition zu einer Stelle führen, werden nur beschriftet, um die Flußzahl anzugeben, falls diese größer als eins ist. Die Flußzahl gibt an, wieviel Marken über den Pfeil fließen, wenn die am Pfeilanfang sitzende Transition schaltet. Auch ein Pfeil, der von einer Stelle zu einer Transition führt, wird mit einer Flußzahl beschriftet, falls diese größer als eins ist. Falls mehrere Pfeile von einer Stelle zu unterschiedlichen Transitionen führen, werden diese Pfeile mit Entscheidungsbedingungen beschriftet. Eine Entscheidungsbedingung ist ein Prädikat, welches einen Anstoß mit Belegungen von operationellen Speicherzellen und von Eingabeangebotskanälen verknüpft. Jeder Zustandsübergangsgraph eines Automaten ist ein Sonderfall eines derartigen Petrinetzes.

5. Beurteilung der "Statecharts" bezüglich ihrer Kommunikationseffizienz

Statecharts wurden 1987 von D. Harel eingeführt [ ]. Er bezeichnet das Konzept der Statecharts als einen *visuellen Formalismus*; es geht also offensichtlich darum, einen Formalismus graphisch zu visualisieren. Daß es sich um einen Formalismus handelt, äußert sich auch in der Einführung des Simulations– und Analysewerkzeugs STATEMATE [ ], dem man die mittels eines Editors gestalteten Statecharts zur formalen Interpretation übergeben kann. Die Arbeiten [ ] befassen sich mit der formalen Semantik der Statecharts.

Zweifellos betrachten es viele Menschen als angenehmer, wenn sie eine Eingabe in einen Formalinterpreter graphisch formulieren dürfen, als wenn sie alphanumerische Ausdrücke übergeben müßten. Da aber ein Formalismus eben doch ein Formalismus bleibt, auch wenn er visualisiert wird, erhebt sich die Frage, ob denn die Visualisierung tatsächlich geeignet ist, beim Betrachter ein intuitives Systemverständnis zu erzeugen. Zur Untersuchung dieser Frage kann man sich nacheinander die Konzepte vornehmen, deren Vereinigung nach Harels Aussage das Wesen der Statecharts ausmacht: Zustandsdiagramme, Tiefe, Orthogonalität und Broadcast–Kommunikation.

Ein *Zustandsdiagramm* ist das angemessene Mittel zur Darstellung des Wertebereichs einer Steuervariablen und der zugehörigen Zustandsübergänge. Dies ist eine unmittelbare Konsequenz des Kriteriums, aufgrund dessen man Operations– und

Steuervariable unterscheiden kann: Eine Steuervariable ist dadurch gekennzeichnet, daß ihr Wertebereich nur definiert werden kann, indem man seine Elemente einzeln aufzählt, und daß die möglichen Zustandsübergänge explizit festgelegt werden müssen, weil sie nicht aus anderen Definitionen abgeleitet werden können.

Mit *Tiefe* ist die Möglichkeit gemeint, jeweils mehrere Zustände zu Superzuständen zusammenfassen zu können. Dies vereinfacht die Darstellung und fördert die Kommunikationseffizienz im Falle von Aussagen, die jeweils für mehrere Zustände in gleicher Weise gelten. Sowohl bezüglich des Ausgabeverhaltens als auch bezüglich des Zustandsübergangsverhaltens von Systemen kann es Aussagen geben, die für mehrere Zustände des Systems in gleicher Weise gelten. Bezüglich des Ausgabeverhaltens lautet eine solche Aussage typischerweise: In allen Zuständen der betrachteten Teilmenge hat eine bestimmte Ausgabevariable vom Displaytyp den gleichen Wert. Bezüglich des Zustandsübergangsverhaltens lautet eine solche Aussage typischerweise: Falls das System in einem Zustand der betrachteten Teilmenge ist und eine bestimmte Eingabe erfolgt, geht das System unabhängig vom aktuellen Zustand in einen bestimmten Folgezustand über. Eingaben, die das System unabhängig vom aktuellen Zustand in einen bestimmten Folgezustand überführen, heißen meistens RESET, EXIT, QUIT oder TIMEOUT.

Mit dem Konzept der *Orthogonalität* soll einerseits das Problem der riesigen Zustandszahlen gelöst und zum anderen die Darstellung von Nebenläufigkeit ermöglicht werden. Der aktuelle Zustand eines Systems ergibt sich jeweils aus der aktuellen Belegung sämtlicher Speicherzellen, die im System vorkommen. Deshalb kann es nebenläufige Belegungsänderungen für unterschiedliche Speicherzellen geben. Nun ist es aber normalerweise unmöglich, die Wertebereiche sämtlicher Zustandsvariablen des Systems in einer Statechart zu erfassen, denn in fast allen praktisch interessierenden Systemen gibt es Zustandsvariable mit äußerst mächtigen Wertebereichen. Zwar kann man durch Kodierung dieser Wertebereiche sekundäre Zustandsvariable mit sehr kleinen, im Extremfall binären Wertebereichen einführen, aber dann bräuchte man eine derart große Zahl sekundärer Zustandsvariablen, daß die praktische Beherrschbarkeit der Statechart trotzdem nicht gegeben wäre. Wenn man jedoch die Unterscheidbarkeit zwischen Operations– und Steuervariablen beachtet, dann erkennt man sofort, daß man die Orthogonalität nur in denjenigen Fällen braucht, in denen das System mehr als eine Steuervariable enthält. Es gibt zwei unterschiedliche Gründe dafür, daß ein System mehrere Steuervariable enthalten kann.

Der eine Grund ist eine Hierarchisierung der Steuerung. In diesem Fall bleibt der Steuerablauf trotz des Vorkommens mehrerer Steuervariablen sequentiell. Die beiden Steuervariablen zweier hierarchisch geordneten Steuerabläufe sind analog zu den Befehlszählern eines Hauptprogramms und eines aus dem Hauptprogramm heraus aufrufbaren Unterprogramms. Im Falle hierarchischer Steuerung gibt es für jeden Ablauf auf einer Hierarchieebene einen eigenen Zustandsgraphen. Die Zustandsgraphen unterschiedlicher Hierarchieebenen sind durch die Aufruf– und Rückkehrereignisse miteinander verkoppelt. Wenn man auch Fälle von Rekursion erfassen muß, steht man vor dem Problem, einen Stackkmechanismus visualisieren

zu müssen. In [ ] wurden geeignet modifizierte Petrinetze vorgestellt, mit denen rekursive Aufruf– und Rückkehrbeziehungen anschaulich dargestellt werden können. Versuche, solche Beziehungen in Statecharts mit Hilfe des Konzepts der *history nodes* darzustellen, lieferten keine Ergebnisse, die an die Kommunikationseffizienz der modifizierten Petrinetze auch nur annähernd herangereicht hätten.

Auch was die Visualisierung von Nebenläufigkeit angeht, kommen die sicher markierten Stellen–Transitionsnetze als einfache Form von Petrinetzen der Gestaltwahrnehmungsfähigkeit des Menschen weit mehr entgegen als die Statecharts mit Orthogonalität. Dies liegt daran, daß bei den Statecharts die Nebenläufigkeit nur durch das Konzept der *broadcast communication* erfaßt werden kann, was sich durch Vorkommen des gleichen Bezeichners an verschiedenen Stellen im Diagramm äußert. Der Betrachter muß also jeweils das gesamte Diagramm daraufhin absuchen, an welchen verschiedenen Orten jeweils der gleiche Bezeichner vorkommt. Bei den Petrinetzen dagegen äußert sich die Nebenläufigkeit in der Topologie und wird deshalb vom Betrachter mühelos erkannt.

Zusammenfassend läßt sich also feststellen, daß man zur kommunikationsfreundlichen Darstellung von Systemen, für die es ein intuitives Systemverständnis gibt und die somit nicht nur bloße Formalismen sind, wesentlich angemessenere Darstellungsmittel zur Verfügung hat als die Statecharts mit den Konzepten Orthogonalität, Broadcast–Kommunikation und Vergangenheitsknoten. Alle intuitiv verständlichen Systeme, die in der Statechart–Literatur als Beispiele gewählt und dargestellt wurden, lassen sich sehr viel verständlicher darstellen, indem man streng zwischen Operations– und Steuervariablen unterscheidet, die Steuervariablen in Petrinetzen erfaßt und die Operationsvariablen als Speicher in ein Aufbaustrukturbild einbringt.

6. Ein Beispiel zur Veranschaulichung des Beschreibungsprinzips

Zum Abschluß dieses Aufsatzes wird ein Beispiel betrachtet, bei dem einer Statechart–Darstellung eine andere Darstellung gegenübergestellt wird, damit sich der Leser selbst ein Urteil über die unterschiedliche Kommunikationseffizienz der beiden Darstellungsformen bilden kann. Als Beispiel wurde die Digitaluhr gewählt, die in verschiedenen Veröffentlichungen über Statecharts oder das Werkzeug STATEMATE beschrieben ist [ ]. Ein System, dessen Darstellung den Rahmen eines Aufsatzes nicht sprengt, kann selbstverständlich kein wirklich komplexes System sein; der Autor hofft aber, daß dieses Beispiel dennoch geeignet ist, dem Leser die Vorteile des hier empfohlenen Beschreibungsprinzips vor Augen zu führen. Leider muß der Leser bezüglich der Statechart–Darstellung der Uhr auf die angegebene Literatur verwiesen werden; eine Aufnahme dieser Darstellung in den vorliegenden Aufsatz war aus Umfangsgründen nicht möglich.

In Bild 1 ist eine Aufbaustruktur der Digitaluhr dargestellt. Das ausschließliche Ziel bei der Gestaltung dieser Struktur war es, dem Leser die Funktion der Uhr möglichst leicht verständlich zu vermitteln. Dazu ist es äußerst hilfreich, eine Aufbaustruktur

zu zeigen, aber es ist nicht erforderlich, daß die Elemente dieser Struktur umkehrbar eindeutig auf die Elemente der realen Aufbaustruktur abbildbar sind. Es muß nur gewährleistet sein, daß die gezeigte Struktur und die reale Struktur bezüglich aller interessierenden Verhaltensmerkmale äquivalent sind. In der Begriffswelt von STATEMATE gibt es die Betrachtung unterschiedlicher verhaltensäquivalenter Aufbaustrukturen auch: Die sogenannten *module charts* sind die Darstellungen der realen Aufbaustrukturen, wogegen in den *activity charts* Aufbaustrukturen dargestellt werden, deren Komponenten im Hinblick auf das intuitive Systemverständnis ohne Berücksichtigung des realen Aufbaus eingeführt werden.

Es gibt in Bild 1 etliche Strukturelemente, die man garantiert auch im realen Aufbau finden wird: Zum einen sind dies die Ein– und Ausgabekanäle, über welche der Benutzer mit der Uhr verbunden ist, und zum anderen sind es die Speicher für die operationellen Zustandsvariablen. Als Eingabekanäle findet man oben links im Bild die vier Drucktasten a, b, c und d. Als Ausgabekanäle findet man die rechts im Bild in einem großen Oval zusammengefaßten Displayfelder für die verschiedenen Anzeigen sowie den akustischen Kanal, der für das Wecken und den Stundenpieps benutzt wird.

Die Speicher für die operationellen Zustandsvariablen sind als schattierte rundberandete Knoten dargestellt. Man findet in der Mitte des Bildes übereinanderliegend die Speicherzellen für die Weckzeit, die aktuelle Uhrzeit, das Datum, den aktuellen Stand der Stoppuhr sowie die Stoppuhrzwischenzeit. Neben diesen Zustandsvariablen mit sehr mächtigen Wertebereichen gibt es auch noch vier binäre operationelle Zustandsvariable, nämlich die Weckerlaubnis, die Stundenpiepserlaubnis, die Stoppuhrlauferlaubnis und die Quellenauswahl für die Stoppuhranzeige.

Die Wertebereiche der bisher aufgeführten operationellen Zustandsvariablen sind aus semantischen Gründen zwangsläufig diskret. Es gibt aber in Bild 1 noch zwei weitere Speicherzellen für operationelle Zustandsvariable, deren Wertebereiche aus semantischen Gründen primär kontinuierlich sind. Es handelt sich um die oben links im Bild befindlichen Speicher für die verbliebene Wartezeit des Timeout–Detektors und die verbliebene Signaldauer des Piepsgenerators. Diese Speicher kann man als Behälter ansehen, die aufgrund eines bestimmten Ereignisses schlagartig eine bestimmte Füllung erhalten und die anschließend kontinuierlich leerlaufen. Die primär kontinuierlichen Wertebereiche dieser Variablen können diskretisiert werden; in diesem Fall stellt man sich unter diesen Speichern Zähler vor, die durch ein Ereignis auf einen bestimmten positiven Anfangswert gesetzt werden und die dann anschließend durch einen Takt abwärtsgezählt werden, bis sie den Wert Null enthalten.

Neben den schattiert dargestellten rundberandeten Knoten, die als Speicherzellen zu interpretieren sind, gibt es in Bild 1 auch etliche rundberandete Knoten, die nicht schattiert sind. Dadurch werden Kanäle symbolisiert. Man kann einen solchen Knoten auch als Symbol für einen Ort ansehen, der am Ende eines Kanals liegt. Die derart symbolisierten Kanäle sind entweder solche, die den Benutzer mit seiner Uhr verbin-

den, oder solche, die von einer aktiven Systemkomponente zu einer anderen aktiven Systemkomponente führen.

Die aktiven Systemkomponenten sind in Bild 1 als Rechtecke dargestellt. Eine aktive Komponente dient jeweils dazu, aus dem, was sie über ihre Eingangskanäle erhält, durch Verknüpfung und Transformation das zu gewinnen, was sie in ihre Ausgabekanäle einspeist. Wenn eine aktive Komponente einen Speicher für einen zeitveränderlichen inneren Zustand enthält, dann ist diese Komponente in Bild 1 schattiert. Eine aktive Komponente, die nicht schattiert ist, wirkt also als Zuordner, der zustandsunabhängig einen funktionalen Zusammenhang zwischen seinen Eingaben und seinen Ausgaben garantiert.

In Bild 1 kommen nur zwei aktive Komponenten vor, die einen zeitveränderlichen inneren Zustand haben. Das eine ist die überwachte Batterie oben rechts im Bild. Diese Batterie muß einen zeitveränderlichen inneren Zustand haben, denn sonst könnte die binäre Batteriezustandsanzeige nicht nach einer bestimmten Betriebszeit vom Wert "Batterie in Ordnung" auf den Wert "Batterie schwach" springen. Die zweite aktive Komponente mit einem inneren Zustand ist die Hauptsteuerung der Uhr; es handelt sich um das große schattierte Rechteck in der linken Bildmitte. Ihr innerer Zustand ist ein Steuerzustand, der darüber entscheidet, welche aktuelle Wirkung ein Tastendruck des Benutzers haben wird.

Jeder Kanal, der zwei rechteckige Komponenten verbindet, ist durch einen rundberandeten Knoten explizit symbolisiert. Wenn der Kanal für die jeweilige Senke ein Anstoßkanal ist, dann ist dies durch ein kleines offenes Dreieck am Senkeneingang symbolisiert. Da die Speicher, die als schattierte rundberandete Knoten dargestellt sind, als passive Systemkomponenten zu betrachten sind, ist jeder Pfeil, der einen solchen Speicher mit einer aktiven Komponente verbindet, auch ein Kanal zwischen zwei Systemkomponenten. Diese Kanäle werden nicht explizit durch eigene Knoten symbolisiert. Ein von einer aktiven Komponete zu einem Speicher führender Pfeil wirkt für den Speicher als Anstoßkanal, über den ein neuer Wert in den Speicher eingeschrieben werden kann. Ein vom Speicher zur aktiven Komponente führender Pfeil ist für die aktive Komponente ein Angebotskanal, auf dem der jeweilige Speicherinhalt angezeigt wird.

Zum Verständnis der Funktion der als Rechtecke dargestellten aktiven Komponenten bedarf es in den meisten Fällen nur eines kurzen Kommentars. Jedesmal, wenn der Benutzer auf eine der vier Drucktasten drückt, werden sowohl der Timeout–Detektor als auch die Ereignisweiche angestoßen. Dieser Anstoß veranlaßt den Timeout–Detektor, den Speicher für die verbliebene Wartezeit auf den Maximalwert zu setzen. Der Benutzer könnte also erreichen, daß die verbliebene Wartezeit niemals den Wert null erreicht, indem er in nicht zu langen Zeitabständen auf eine Drucktaste drückt. Wenn der Speicher für die verbliebene Wartezeit aber einmal leergelaufen ist, dann erzeugt der Timeout–Detektor an seinem Ausgang ein Anstoßereignis für den Konfliktentscheider. Dieser Konfliktentscheider wird gebraucht, weil es möglich ist, daß der Benutzer genau in dem Augenblick auf eine Drucktaste drückt, wenn der Ti-

meout–Detektor das Ende des Zeitintervalls meldet. In diesem Falle erhält der Konfliktentscheider gleichzeitig zwei Anstöße, nämlich einen vom Timeout–Detektor und einen weiteren von der Ereignisweiche. Die Funktion des Konfliktentscheiders besteht einfach darin, alle Anstöße, die nicht gleichzeitig kommen, unverändert an die zentrale Steuerung weiterzugeben, und im Falle des gleichzeitigen Eintreffens zweier Anstöße die Timeoutmeldung zu ignorieren und nur das Tastendruckereignis weiterzumelden.

Die Ereignisweiche wird gebraucht, weil nicht jedes Tastendruckereignis an die zentrale Steuerung geleitet werden soll. Solange der Piepsgenerator ein akustisches Signal erzeugt, soll jedes beliebige Tastendruckereignis ausschließlich dazu benutzt werden, den Piepsgenerator zu stoppen. Die Möglichkeit, das Piepsen durch jeden beliebigen Tastendruck sofort abstellen zu können, ist äußerst wünschenswert, denn man muß ja immer damit rechnen, daß die Uhr in äußerst unpassenden Situationen anfängt zu piepsen. Wenn das Piepsen nur mittels einer ganz bestimmten Taste abgestellt werden könnte, würde es möglicherweise unnötig lange dauern, bis der Benutzer die richtige Taste gefunden hat. Die Ereignisweiche lenkt deshalb jedes Tastendruckereignis zum Piepsgenerator, solange der Speicher für die verbliebene Piepszeit noch nicht leergelaufen oder auf null gesetzt worden ist.

Der Piepsstart–Detektor hat lesenden Zugriff zu verschiedenen Speichern für operationelle Zustandsvariable. Wenn weder die Weckerlaubnis noch die Stundenpiepserlaubnis gegeben sind, braucht der Piepsstart–Detektor die anderen Variablen nicht anzusehen, denn in diesem Fall wird er ja keinen Piepsstart produzieren. Wenn die Weckerlaubnis gegeben ist, muß der Piepsstart–Detektor dauernd die aktuelle Uhrzeit mit der eingestellten Weckzeit vergleichen, damit er den Augenblick nicht verpaßt, in dem diese beiden Zeiten gleich werden. Wenn die Stundenpiepserlaubnis gegeben ist, muß der Piepsstart–Detektor dauernd auf die aktuelle Uhrzeit schauen, damit er jedesmal, wenn eine volle Stunde erreicht ist, den Piepsgenerator starten kann.

Da es fünf Quellen gibt, aus denen die numerische Anzeige der Uhr gespeist werden kann, muß eine Quellenselektion erfolgen. Die Aufbaustruktur in Bild 1 enthält zwei Selektoren, von denen der eine aus vier Quellen und der andere aus zwei Quellen auswählt. Der aus zwei Quellen auswählende Selektor wird dazu benutzt, entweder den aktuellen Stand der Stoppuhr oder die gespeicherte Stoppuhr–Zwischenzeit zur Ausgabe zu lenken.

Die einzigen beiden aktiven Komponenten in Bild 1, deren Funktion man nicht durch einen einfachen Kommentar vollständig vermitteln kann, sind die beiden links im Bild liegenden Steuerungen. Die genaue Funktion dieser beiden Steuerungen wird durch die Zustandsgraphen beschrieben, die in Bild 2 dargestellt sind. Die Stoppuhrsteuerung wird nur angestoßen, wenn die zentrale Steuerung ihren Anstoß unverändert zur Stoppuhrsteuerung weitergibt. Dies geschieht nur, wenn sich die zentrale Steuerung in einem ganz bestimmten Zustand befindet. Dieser Zustand liegt rechts unten im Zustandsgraphen der Zentralsteuerung.

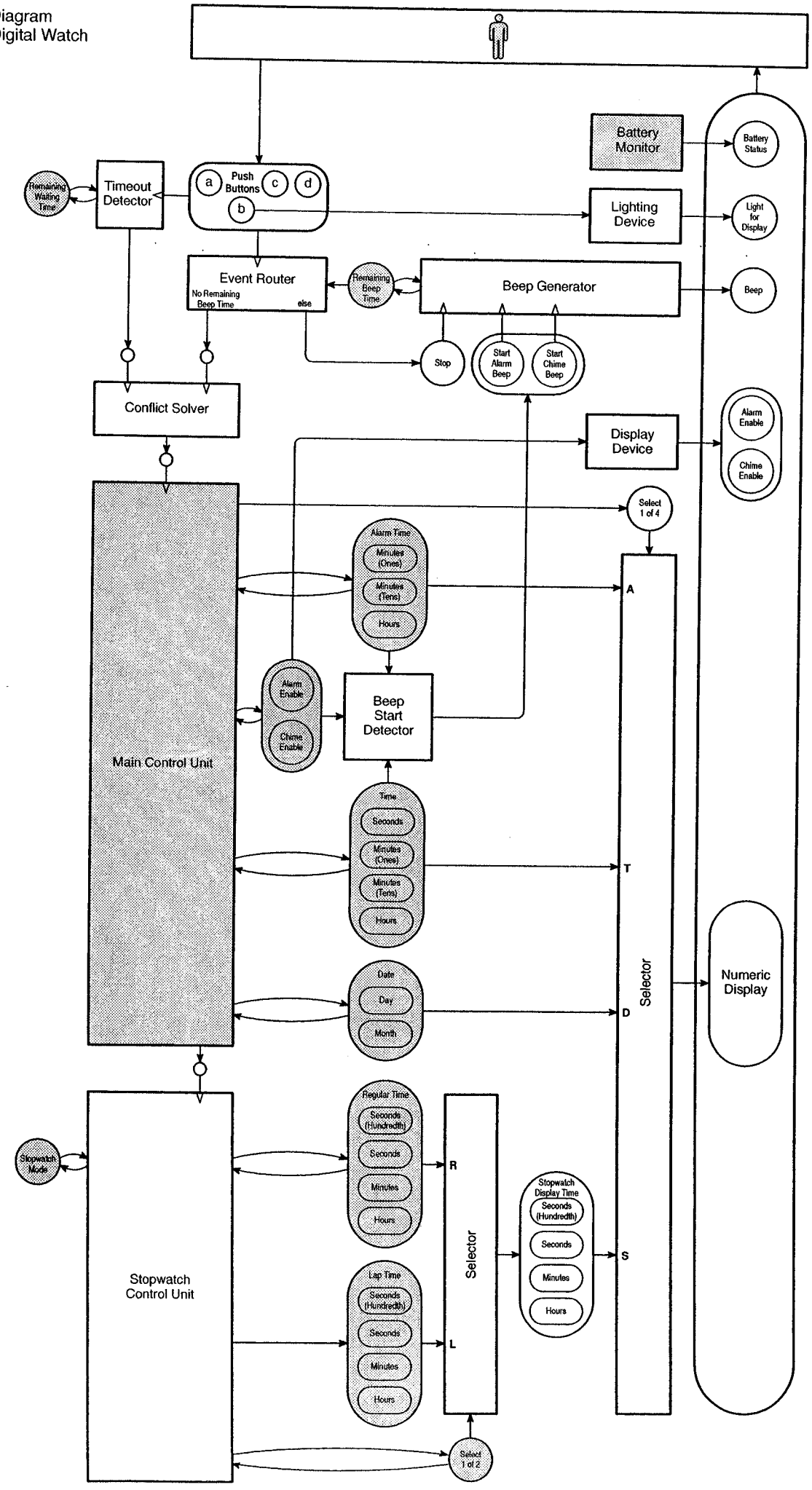
Die Zustandsgraphen in Bild 2 sind als Petrinetze dargestellt. In die kreisförmigen Zustandsknoten sind jeweils bestimmte Variablenbelegungen eingetragen, die eindeutig mit dem jeweiligen Zustand verbunden sind. Die Einträge in den Zustandsknoten des Zustandsgraphen der zentralen Steuerung beziehen sich auf die Ansteuerung des großen Selektors in Bild 1, der jeweils eine aus vier Quellen zum numerischen Anzeigefeld durchschalten muß. Die Variable, deren jeweilige Belegung hier dargestellt ist, ist selbst keine Zustandsvariable, obwohl ihre Belegung vom Zustand der zentralen Steuerung abhängt. Es handelt sich um die unmittelbar über dem großen Selektor in Bild 1 liegende Quellenauswahlvariable, die als unschattierter kreisförmiger Knoten dargestellt ist.

Die beiden Variablen, deren zustandsabhängige Belegung in den Zustandsknoten des Graphen der Stoppuhrsteuerung eingetragen ist, sind Zustandsvariable. Man findet sie in Bild 1 als schattierte kreisförmige Knoten, mit denen die Stoppuhrsteuerung verbunden ist. Es handelt sich zum einen um die Lauferlaubnis und zum anderen um die Quellenauswahl für die Stoppuhranzeige. Obwohl die Stoppuhrsteuerung in Bild 1 ein nichtspeichernder Zuordner ist, kann man doch von den vier möglichen Zuständen der Stoppuhrsteuerung sprechen, indem man die beiden Operationsvariablen jeweils als binäre Komponenten eines zweistelligen Zustandscodewortes auffaßt.

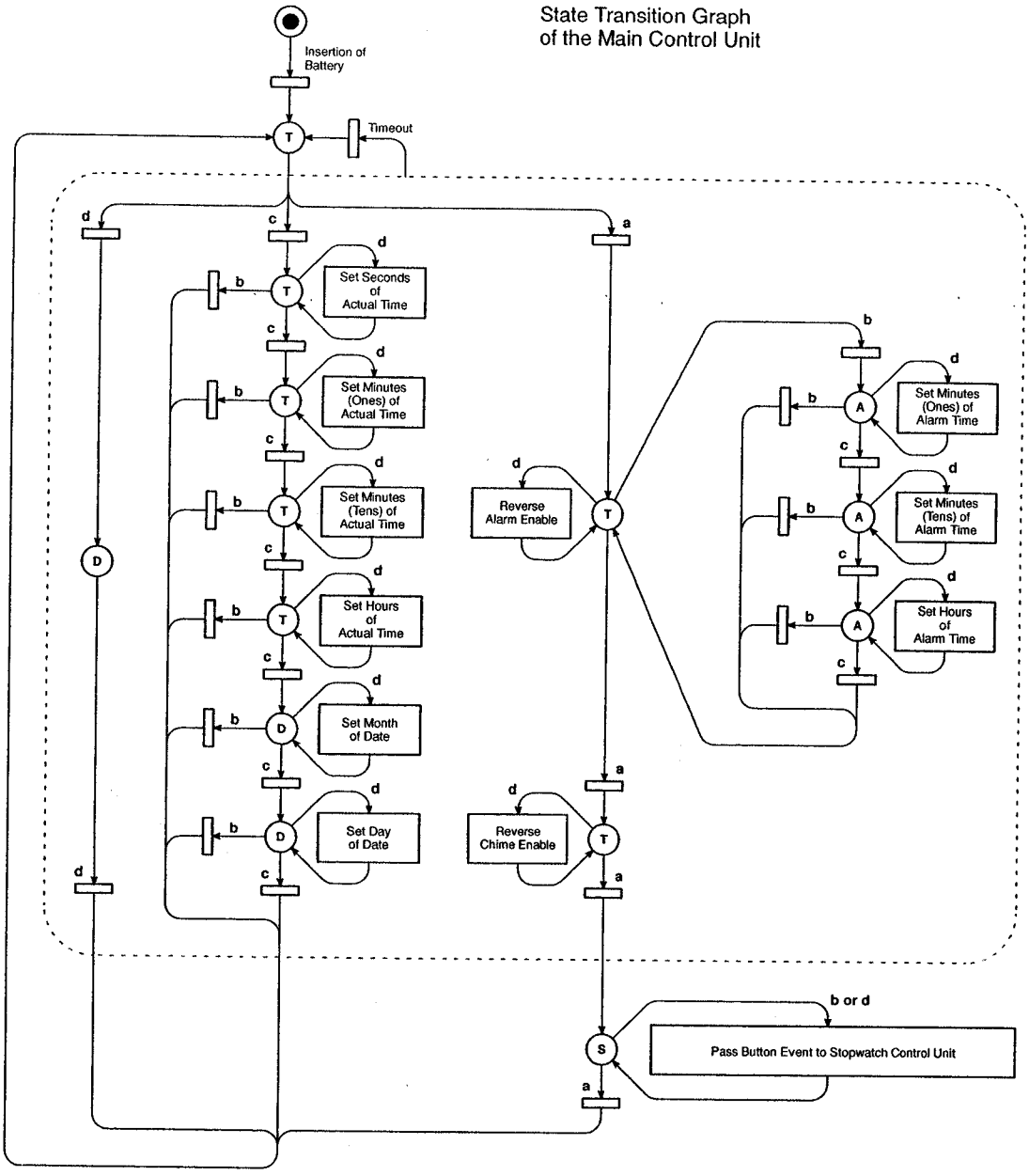
In den Graphen in Bild 2 sind nur diejenigen Tastendruckereignisse eingetragen, die eine relevante Wirkung haben. Eine solche Wirkung besteht immer in einem Zustandsübergang. Falls es sich um einen Zustandsübergang bei Operationsvariablen handelt, ist dieser Übergang durch einen entsprechenden Eintrag in der durch das Tastendruckereignis ausgelösten Transition charakterisiert. Falls der Tastendruck nur einen Steuerzustandsübergang bewirkt, ist die zugehörige Transition unbeschriftet; es handelt sich um die vielen kleinen Rechtecke in den Graphen.

Für alle Zustände, die sich innerhalb des gestrichelt umrandeten Teilnetzes befinden, ergibt sich als Wirkung des Timeout–Ereignisses ein Übergang in den gleichen Grundzustand. Der mit "Timeout" beschriftete Zustandsübergang steht stellvertretend für zwölf Zustandsübergänge, weil sich innerhalb des gestrichelt umrandeten Teilnetzes zwölf Zustandsknoten befinden, von denen jeweils ein solcher Übergang ausgehen kann. Die Möglichkeit, durch die Bildung von Zustandsteilmengen die Darstellung zu vereinfachen, wurde bereits im Abschnitt 2 über Statecharts erwähnt. In der Literatur über Statecharts wird die Möglichkeit der Zustandsteilmengenbildung als Konzept der *Tiefe* bezeichnet.

Block Diagram
of the Digital Watch



State Transition Graph
of the Main Control Unit



State Transition Graph
of the Stopwatch Control Unit